Bachelor's Thesis (Academic Year 2021)

Qubit Allocation For Heterogeneous Quantum Computing

Keio University, Faculty of Environment and Information Studies
Makoto Nakai

Abstract of Bachelor's Thesis - Academic Year 2021

Qubit Allocation For Heterogeneous Quantum Computing

Quantum computers are theoretically capable of solving some intractable problems for classical computers including the factoring problem and the digital quantum simulations in a polynomial time, and large-scale quantum computing is the key to perform computation that even a supercomputer cannot handle. Two approaches for large-scale quantum computing have been proposed. One is to build a single large quantum processor, and the other is to perform quantum computing over more than one quantum processors that are connected via communication links. The later approach is considered as more practical because it requires less number of qubits in each quantum processor.

However, only few works have investigated the procedure to perform distributed quantum computing in the real world, especially how to convert a user program which includes the quantum circuit to the executable form onto distributed quantum computers.

This work proposes the heuristic optimization algorithm for qubit allocation for distributed quantum computing which aim to shorten the total execution time of the given quantum circuit, and demonstrated that the qubit allocation by this algorithm achieve reduction of the total execution time compared to the random qubit allocation by numerical simulation.

Keywords :
1. Quantum computing, 2. Distributed system, 3. Task allocation, 4. Simulated Annealing

Keio University, Faculty of Environment and Information Studies
Makoto Nakai

Contents

List of Figures

Chapter 1

Introduction

1.1 Background

In 1982, the idea of quantum computing was proposed by Richard Feynman [1] with the intention of simulating quantum system whose dimension grows exponentially if the number of particles increase. In 1990s, quantum computers are proved to be theoretically capable of solving some intractable problems for classical computers including the factoring problem [2] and the digital quantum simulations in a polynomial time [3], and large-scale quantum computing is the key to perform computation that even a supercomputer cannot handle. The hardware for quantum computing has been developed using various physical architecture such as superconducting system [4], ion trap [5], NV diamond [6].

Two approaches for large-scale quantum computing have been proposed. One is to build a single large quantum processor [7], and the other is to perform quantum computing over more than one quantum processors that are connected via communication links [8].

However, unlike the previous works about the system for quantum computing on a single quantum processor [9, 10], only few works have investigated the procedure to perform distributed quantum computing in the real world [11], especially how to convert an user program which includes the quantum circuit to the executable form onto distributed quantum computers.

1.2 Research Contribution

The main contribution of this project is the heuristic optimization algorithm for qubit allocation problem for distributed quantum computing, which determines which qubits and associated quantum gates will be executed on which quantum processor. This work adopts the total execution time as the optimization criteria, because the algorithm for qubit allocation with emphasis on total fidelity of the output quantum state can be solved by the existing quantum compilation techniques.

The experiment in this thesis was performed by the self-made distributed quantum computing simulator called heqsim (Heterogeneous Quantum Computing Simulator), which will be explained in more details in Chapter 5.

1.3 Thesis Structure

This thesis is constructed as follows. Chapter 2 provides the fundamentals knowledge of quantum information, distributed system, and distributed quantum computing. In chapter 3 provides the previous researches in the fields of distributed quantum computing that are directly related to this work. Chapter 4 describes the details of the qubit allocation problem for distributed quantum computing and propose its heuristic solution. Chapter 5 explains the detail of heqsim, the distributed quantum computing simulator used in this thesis. Chapter 6 explains the setting of the experiments and their results. Chapter 7 discusses the validity, and the significance, and the future works of this research.

Chapter 2

Background

2.1 Quantum Computing

In this chapter, the author will provide fundamental knowledge about quantum computing, including the concepts of quantum bits, quantum gates, quantum circuit and quantum compilation.

2.1.1 Quantum Bit

A classical bit has two different states, which are 0 and 1. Instead, those of a quantum bit (or **qubit** in short) are $|0\rangle$ and $|1\rangle$, each of which can be described as a vector. For example

$$|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

$$|1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

. The state of a single qubit $|\psi\rangle$ can be described as follows.

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle \, (\alpha, \beta \in \mathbb{C}, |\alpha|^2 + |\beta|^2 = 1)$$

. After the operation called measurement, the quantum state would be collapsed into either 0 or 1. The measurement probability of 0 is $|\alpha|^2$ and that of 1 is $|\beta|^2$. In other words, a single qubit can take both states probabilistically at the same time. For instance, a qubit can be

$$|\psi\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle \tag{1}$$

which can be 50% 0 and 50% 1.

2.1.2 Bloch sphere

Because $|\alpha|^2 + |\beta|^2 = 1$, the notation of a single qubit state can be represented like this.

$$|\psi\rangle = e^{i\gamma}(\cos\frac{\theta}{2} + e^{i\phi}\sin\frac{\theta}{2})(\gamma, \phi, \theta \in \mathbb{R}) \tag{2.1}$$

.

Because $e^{i\gamma}$ is just a global state, it can be ignored and the same state can be rewritten like this.

$$|\psi\rangle = \cos\frac{\theta}{2} + e^{i\phi}\sin\frac{\theta}{2}(\phi, \theta \in \mathbb{C}) \tag{2.2}$$

Because the equation above has two parameters, any pure single qubit state can be considered as a point on the surface and its geometric representation is called **Bloch sphere**.

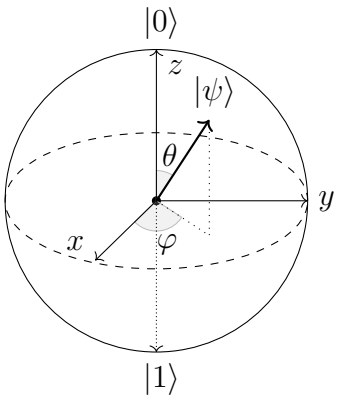


Figure 2.1: Bloch Sphere

2.1.3 Multi-Qubit State

The quantum state for multi-qubits is a **tensor product** of a state vector of each qubit. The general notation of two qubit state is

$$|\psi\rangle = (\alpha|0\rangle + \beta|1\rangle) \otimes (\gamma|0\rangle + \delta|1\rangle) \tag{2.3}$$
$$= \alpha\gamma|00\rangle + \alpha\delta|01\rangle + \beta\gamma|10\rangle + \beta\delta|11\rangle \tag{2.4}$$
$$(\alpha, \beta, \gamma, \delta \in \mathbb{C}, |\alpha|^2 + |\beta|^2 + |\gamma|^2 + |\delta|^2 = 1) \tag{2.5}$$

.

For example, the state $|00\rangle$ is equal to

$$\begin{bmatrix} 1 \\ 0 \end{bmatrix} \otimes \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{2.6}$$

.

However, some quantum states such as

$$|\psi\rangle = \frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle \tag{2.7}$$

cannot be decomposed into quantum state of each qubit. These special quantum states are called **entangled** states.

2.1.4 Quantum Gates

In this section, the author will talk about "logical gates" for quantum computers, which are called **quantum gates**.

I gate

I gate is equal to the 2x2 identity matrix, which is

$$I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{2.8}$$

.

For example,

$$I|0\rangle = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} = |0\rangle \tag{2.9}$$

$$X|1\rangle = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} = |1\rangle \tag{2.10}$$

.

X gate

X gate flips the logical value of a qubit.

$$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \tag{2.11}$$

.

For example,

$$X|0\rangle = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} = |1\rangle \tag{2.12}$$

$$X|1\rangle = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} = |0\rangle \tag{2.13}$$

.

Y gate

Y gate flips the logical value of a qubit and add an imaginary number.

$$Y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix} \tag{2.14}$$

.

For example,

$$Y|0\rangle = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ i \end{bmatrix} = i|1\rangle \tag{2.15}$$

$$Y|1\rangle = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} -i \\ 0 \end{bmatrix} = -i|0\rangle \tag{2.16}$$

.

Z gate

Z gate flips the phase of $|1\rangle$

$$Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \tag{2.17}$$

.

For example,

$$Z|0\rangle = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} = |0\rangle \tag{2.18}$$

$$Z|1\rangle = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ -1 \end{bmatrix} = -|1\rangle \tag{2.19}$$

.

H gate

H gate creates superposition.

$$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \tag{2.20}$$

.

For example,

$$H|0\rangle = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \tag{2.21}$$

$$H|1\rangle = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ -1 \end{bmatrix} = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \tag{2.22}$$

.

Rx gate

An Rx gate rotate the given quantum circuit on the x-axis of the Bloch sphere.

$$Rx(\theta) = \begin{bmatrix} \cos\frac{\theta}{2} & -i\sin\frac{\theta}{2} \\ -i\sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{bmatrix} \tag{2.23}$$

.

For example,

$$Rx(\theta)|0\rangle = \begin{bmatrix} \cos\frac{\theta}{2} & -i\sin\frac{\theta}{2} \\ -i\sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} \cos\frac{\theta}{2} \\ -i\sin\frac{\theta}{2} \end{bmatrix} = \cos\frac{\theta}{2}|0\rangle - i\sin\frac{\theta}{2}|1\rangle \tag{2.24}$$

$$Rx(\theta)|1\rangle = \begin{bmatrix} \cos\frac{\theta}{2} & -i\sin\frac{\theta}{2} \\ -i\sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} -i\sin\frac{\theta}{2} \\ \cos\frac{\theta}{2} \end{bmatrix} = -i\sin\frac{\theta}{2}|0\rangle + \cos\frac{\theta}{2}|1\rangle \tag{2.25}$$

.

Ry gate

An Ry gate rotate the given quantum circuit on the y-axis of the Bloch sphere.

$$Ry(\theta) = \begin{bmatrix} \cos\frac{\theta}{2} & -i\sin\frac{\theta}{2} \\ -i\sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{bmatrix} \tag{2.26}$$

.

CNOT gate

A CNOT gate involves two qubits, one is called **controlled qubit** and the other is called **target qubit**. If the controlled qubit is 1, the bit value of the target qubit is flipped.

$$CNOT = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{2.27}$$

.

For example,

$$CNOT_{0,1}|10\rangle = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} = |11\rangle \tag{2.28}$$

$$CNOT_{0,1}|11\rangle = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} = |10\rangle \tag{2.29}$$

.

Quantum gate for multi-qubit system

Just like quantum state of a multi-qubit system, the composited quantum gates is the tensor product of quantum gates that are applied on each qubit.
For example,

$$X_0 \otimes X_1 = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \otimes \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \tag{2.30}$$

.

Therefore,

$$X_0 X_1 |00\rangle = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} = |11\rangle \tag{2.31}$$

$$X_1 |00\rangle = I_0 X_1 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \otimes \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{2.32}$$

$$I_0 X_1 |00\rangle = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} = |01\rangle \tag{2.33}$$

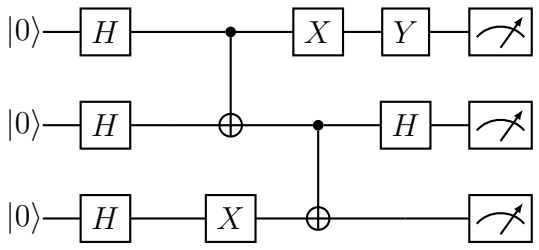
.

Measurement

If a person measure a single qubit, he would get either 0 or 1 and that operation completely destroys the quantum state. It will return 00, 01, 10, 11 in the case of two qubits.

2.1.5 Quantum Circuit

Here is the example of a quantum circuit.



.

Each horizontal line represents each qubit and the square boxes that contain alphabets mean single quantum gates. The sign which involves a vertical line means a CNOT gate, and the box on the most right side indicates measurement.

2.1.6 Bell state

In the chapter 1.3, the author mentions about a special type of quantum states called entangled states, and there are four specific quantum states called "Bell state", which are

$$|\Phi^+\rangle = \frac{|00\rangle + |11\rangle}{\sqrt{2}} \tag{2.34}$$

$$|\Phi^-\rangle = \frac{|00\rangle - |11\rangle}{\sqrt{2}} \tag{2.35}$$

$$|\Psi^+\rangle = \frac{|01\rangle + |10\rangle}{\sqrt{2}} \tag{2.36}$$

$$|\Psi^-\rangle = \frac{|01\rangle - |10\rangle}{\sqrt{2}} \tag{2.37}$$

.

2.1.7 Quantum compiler

Quantum compiler is a software that finds (near-) optimal mapping between the qubits in the program onto physical qubits on a quantum processor, which are constrained by their limited connectivity. Also, its solution has to reduce the number of gates applied on those qubits, which ends up with less physical noise on the actual quantum state. In this chapter, the author will explain the algorithm to find the (near-) optimal qubit allocation presented in [12].

Here is the procedure of quantum compilation.

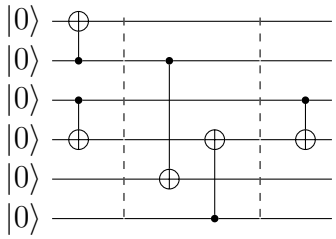


Figure 2.2: An example of quantum circuit to be compiled

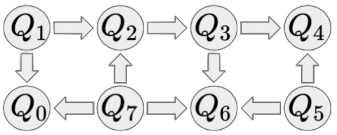


Figure 2.3: The connectivity constraints of qubits on a quantum processor

Circuit Partitioning into Layers

First of all, the given quantum circuit is partitioned into several layers and a layer l_i and all the quantum gates in a single layer l_i do not share any qubits. Also, the number of the depth of a circuit is same as that of layers in the same circuit.

After that, the qubit mapping is proposed in each layer l_i and it does not have to correspond with that of the previous layer l_{i-1}. Instead, a new layer of SWAP gates π_i is inserted before the quantum compiler goes onto the next layer. In other words, the sequence of layers will be denoted as $l_0\pi_1 l_1 \pi_2 l_2$.

Determining the Qubit Mapping

The quantum compiler determines the mapping relationship between program qubits onto physical qubits on an actual quantum processor for each layer l_i, which is denoted as $\sigma_j^i : \{q_0, q_1, q_2, \ldots q_{n-1}\} \rightarrow \{Q_0, Q_1, Q_2, \ldots Q_{m-1}\}$ (j is a unique id for that relationship, n is the number of qubits on the given program, and m is the number of physical qubits on a quantum processor). This relationship has to satisfy the CNOT-constraints on that

hardware and the initial mapping is same of that of the previous layer, in other words, $\sigma^{i-1} = \sigma_0^i$. The goal of this step is to find σ^i, which is the (sub-)optimal mapping relationship by adding minimum number of new operations by using an algorithm A^* search.

A* Search

A^* search is a state-space-based search algorithm that can be applied to states whose relationships have a tree structure. The cost of the next node from the root to the node x is $f(x)$, and $g(x)$ is the cost of the path from the root to the current state, and $h(x)$ is the additional cost of the link between the current state and the next state.

Therefore,

$$f(x) = g(x) + h(x) \tag{2.38}$$

.

In the case of quantum compilation, the cost of the root state, which is the initial qubit mapping is 0. If the cost for the current state is $f(x)$, $g(x)$ is that of the previous mapping, and the $h(x)$ is the cost for additional layers.

If the current mapping σ_j^i is followed by the next mapping σ_h^i, the cost of the current mapping would be

$$f(\sigma_j^i) = f(\sigma_j^i) + 7 \times (\text{number of SWAP gates}) \tag{2.39}$$

.

The equation above contains 7, because a single SWAP gate requires 7 elementary gates.

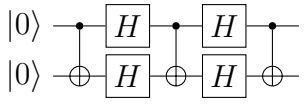


Figure 2.4: 7 elementary gates to implement a SWAP gate

If you assign physical qubits Q_c as a control qubit and Q_t as a target qubit for a CNOT gate $g \in l_i$. If the shortest directed path between Q_c and Q_t is denoted as $\hat{p}$, the additional gate cost for the CNOT gate g in the current mapping σ_j^i would be

$$h(g, \sigma_j^i) = 7 \times (\hat{p} - 1) \tag{2.40}$$

.

Therefore, the gate cost of additional layer l, which is $h(\sigma_j^i)$ is

$$h(\sigma_j^i) = \max_{g \in l} h(g, \sigma_j^i) \tag{2.41}$$

.

Further Optimization

In order to perform further optimization, the quantum compiler can incoroperate the information of not only the current layer, but also that of the next layer. For example, the cost of the current layer can be

$$h(\sigma_j^i) = \max_{g \in l_i \cup l_{i+1}} h(g, \sigma_j^i) \tag{2.42}$$

.

2.2 Distributed Computing

Distributed computing is the study of distributed system, which is a collection of several processors that are connected via network in order to solve a problem whose scale is much larger than what an individual processor can handle. This type of computation involves message-passing between two physically separated processors to communicate and cooperate with each other by using pure HTTP, RPC, or message queues. [13]

2.2.1 Characteristics of Distributed System

Distributed system has the following characteristics, which are,

- No common physical clock

  This is the key feature of distributed system because the clock of each processor runs at a different rate, so no clock can keep synchronized even after a single physical clock cycle. Instead, distributed system depends on logical clock, which is common time platform for the whole system.

- No shared memory

  Each processor in a distributed system has its own memory space, rather than the common physical memory. This feature indicates that a distributed system does not share its global state.

- Geographical separation

  The processors in a distributed system is located in different places, but they do not have to communicate with wide area network (WAN). Actually, the network of workstations (NOW) and the cluster of workstations (COW) are becoming increasing popular because companies can easily purchase cost-efficient, high-speed, and ready-made processors.

- Autonomy and heterogeneity

  A distributed system can work together even if it contains various processors which have different size, speed and operating system as long as they cooperate with one another. This situation is regarded as "loosely-coupled".

2.2.2 Advantages Over Computation by a Single Processor

Performing computation in a distributed manner provides the following advantages.

- Computation by more than one entity

  Applications such as money transfer (client-server) and reaching consensus among parties that are geographically separated (peer-to-peer) require information processing system that each processor can work together.

- Resource sharing

  Resource such as data in databases cannot be replicated because it is impossible or at least cost effective. Furthermore, allocating all the resource in just a single server is also not practical because the whole application would become unavailable if the server fails. In order to solve these potential problems, the whole dataset is usually partitioned into several servers so that it can achieve more rapid access and higher reliability.

- Access to geographically remote data and resources

  Copying the whole dataset to every site is not desirable due to not only its predicted high cost, as I mentioned in the previous section, but also too sensitive. Therefore, these large amount sensitive information, like user information collected by multi-national cooperations are stored only in their central data centers and their oversea branches are only allowed to query them.

- Enhanced reliability

  Distributed system offers increased reliability due to its ability to replicate resource and achieve simultaneous execution of its given tasks. Also geographically distributed resources are highly unlikely crash or malfunction at the same time under normal circumstance. This eliability entails several aspects.

  - Availability

    Resource become accessible at all times

  - Integrity

    The value and state of the resources should be always correct, especially users get concurrent access to those resources.

  - Fault-tolerance

    Distributed system should be able to recover from its failure such as one of its server accidentally shutting down

- Increased performance / cost ratio

  By resource sharing and access to geographically distant data, the performance and cost ratio of distributed system will improve more than using special parallel machines, this is particularly true of the NOW (network of workstation) setting.

2.2.3 Task Allocation Problem

Here are some definitions of the task allocation problem in distributed system. [14] Given a distributed system $G =< V, E >$, where V is the set of nodes and E is the set of communication link between two different nodes, i.e. $\forall v_i.v_j \in V, \exists < v_i, v_j > \in E$. The set of the resources in a_i is R_{a_i} and that required by the task t is R_t.

1. $R_t \subseteq \bigcup_{\forall a_i \in A_t} R_{a_i}$.

2. The objective should be either minimizing the execution time [15] or maximizing reliability [16].

3. The nodes in A_t can execute the allocated task under the constraint of the network structure $\forall a_i, a_j \in A_t \Rightarrow P_{ij} \subseteq E$ where P_{ij} denotes the path between a_i and a_j.

Task allocation is known to be a NP-problem [17].

2.2.4 Distributed Task Allocation Algorithms

Due to the fact that task allocation problem is classified as NP-hard, many works have proposed heuristic functions for distributed task allocations in various settings. Here, the author is going to introduce a simple objective function presented in the paper [18], which is this work is based on. It is assumed that a multicomputer system consists of N heterogeneous processors with some amount of computational power and the size of memory. Also, the given program includes M communicating tasks, which is the nodes in the task interaction graph $G(V, E)$ (V is M communicating tasks and E is a set of communication relationship between two tasks).

The objective of this task allocation is to minimize the total execution time. In other words, the author has to come up with the optimal allocation A, which $A(i) = p$ indicates that the task i is allocated to the processor p and $TASKS_p$ is a group of tasks that are allocated to a processor p. The total execution in the heterogeneous computing cluster is same as the execution time in the most heavily loaded processor. Two main types of costs should be considered. One is the execution cost. The execution load in the processor p is the cost of processing all the tasks that are assigned to p for the allocation A.

Suppose C_{ip} is the cost of processing the task i on the processor p, then the total execution cost on the processor p is

$$\mathrm{EXEC_p} = \sum_{task \in \mathrm{TASKS_p}} \mathrm{C_{task,p}} \tag{2.43}$$

.

The other cost is the communication cost, which can be calculated by the following formula.

$$\mathrm{COMM_p} = \sum_{task \in \mathrm{TASKS_p}} \sum_{(i=j),(i,j) \in \mathrm{E}, A(j) \neq p} \mathrm{d_{ij}} * \mathrm{cc_{avg}} \tag{2.44}$$

.

d_{ij} is the data sent between two communicating tasks between i, j and cc_{avg} is the average amount of transferring a data unit through the network transmission media.

Therefore, the total cost for the processor p is

$$\mathrm{COST_p} = \mathrm{EXEC_p} + \mathrm{COMM_p} \tag{2.45}$$

. Because the total execution time is equal to the execution of the most heavily loaded processor, the total execution time can be described as following.

$$\mathrm{COST} = \max\{\mathrm{COST_p} \,|1 \leq \mathrm{p} \leq \mathrm{n}\} \tag{2.46}$$

.

Therefore, the object function is

$$\min \text{ COST} \tag{2.47}$$

.

2.2.5 Models of Process Communication

There are two basic models of process communication. One is *synchronous* communication, which the sender process blocks its execution until it receives "acknowledgement " (or "ack" in short) from the receiver, which indicates that the receiver accepted the delivered message. In other words, the sender and receiver synchronize with each other, in order to adjust their timings to cooperate.

On the other hand, the other model, which is *asynchronous* execution does not require synchronization between the sender and the receiver. Therefore, the sender executes its following task right after it delivers its message to the receiver.

2.3 Distributed Quantum Computing

Performing quantum computation on a cluster of middle-sized quantum processors is an easier approach compared to build a large single processor due to its lower physical requirement to build each processor [19].
This chapter explains several components that consists of distributed quantum computing system.

2.3.1 Distributed Quantum Algorithms

Distributed version of Shor's algorithm [20] and Grover's algorithm [21], both of which takes remote CNOTs (which will be discussed in a later subsection) were presented in 2004 and 2012, respectively. However, the quantum circuits shown in these studies are physical ones, in other words, something that will be implemented directly on physical hardware, not something that users will implement in their programs.

2.3.2 Quantum Compiler for Distributed Quantum Computing

Quantum compiler is a software to optimize a quantum circuit defined in the given program to reduce the number of quantum gates and alleviate the effect of physical noise on that quantum state, and map that to the real hardware so that it will satisfy the connectivity constraint on that hardware [9]. Just like the case of a local quantum software, distributed quantum computer also needs its distributed version in order to convert some CNOT gates into remote CNOTs and optimize both its execution time and the number of quantum gates in total, especially that of non-local operations. This work focuses on the methodology to optimize the total execution time over the distributed quantum computing setting.

2.3.3 Gate-Teleportation-Based Non-Local CNOT

Non-local operations are controlled operations between two qubits on two different processors there are three main approach to achieve them. The first operation is the gate teleporation approach, and is also called non-local quantum gates [22] and telegate [23]. Here is the quantum circuit that achieve this operation.

Suppose that a person would like to apply a non-local CNOT between $|a_0\rangle$ on one processor and $|b_0\rangle$ on the other processor. In order to do this, he has to prepare a new bell pair between these two processors.

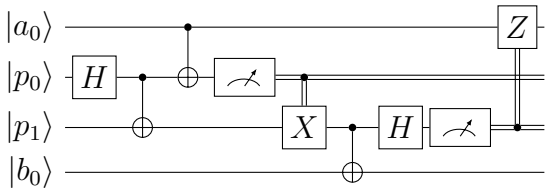


Figure 2.5: Quantum circuit for a non-local CNOT gate

2.3.4 Quantum Teleportation

Unlike classical communication, quantum states cannot be just copied and transmit to other nodes due to the no-cloning theorem, which forbids duplication of any quantum state. However, a method called quantum teleportation [24] was proposed, which overcomes the restriction and allows sender to transmit single qubit state to a distant location.

This method requires both the single qubit state and a new Bell pair, and also the sender have to prepare two qubits and the receiver have to prepare one qubit. After applying a CNOT gate and an H gate in the figure above, the sender have to measure both qubits and send those measurement results over the classical network. After the receiver get those measurement results and apply some quantum gates if the measurement results of corresponding qubits on the sender's side are 1, in order to correct on the quantum state on the receiver's side.

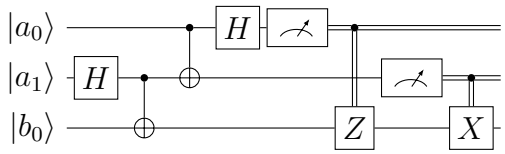


Figure 2.6: Quantum circuit for quantum teleportation

2.3.5 Quantum-Teleportation-Based Non-Local CNOT

The second approach for performing a non-local CNOT gate is based on quantum teleportation mentioned in the previous section. This approach assumes that every quantum processor has something called "communication qubits", which is a qubit that serves for communication purpose, unlike those used for computation purpose, which are called "data qubits".

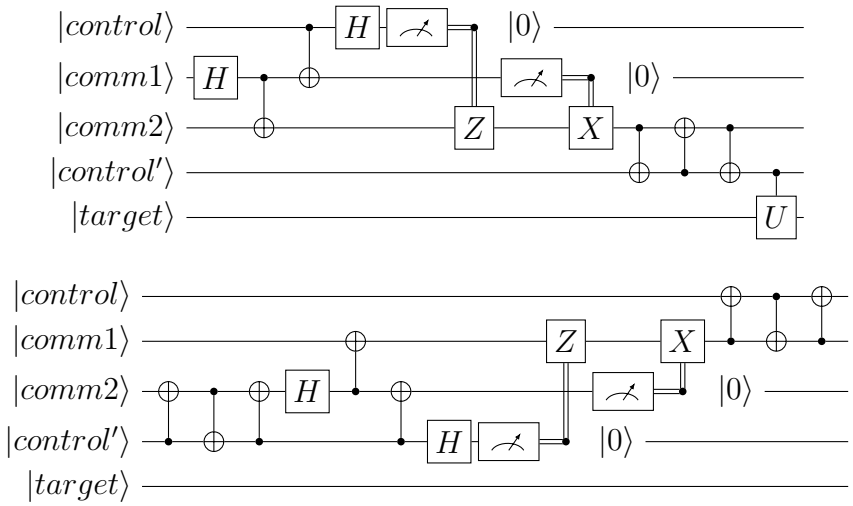


Figure 2.7: The full quantum circuit for a teledata non-local controlled-U gate

2.3.6 Data-Qubit-Swapping-Based Non-Local-CNOT

The last approach for the non-local CNOT gate aims to sort qubits in the quantum processors so that each CNOT gates would be executed on the neighboring processors. (the paper [25] assumes linear topology)

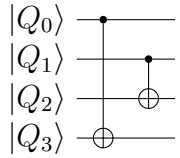


Figure 2.8: The quantum circuit before the data qubit swapping occurs

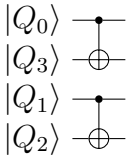


Figure 2.9: The quantum circuit after the data qubit swapping occurs

Algorithm 1 Algorithm for Data-Qubit Swapping

Input: n-qubit circuit layer L with mod(n, 4) = 0 and $\frac{n}{2}$ CNOTs
Output: layer L with each CNOT operating on neighbor qubits

function SORT(L)
  if $\exists$ CNOT(q_i, q_j) with $i, j \leq \frac{n}{2}$ **then**
    // $\exists$ CNOT(q_k, q_l) with $k, l > \frac{n}{2}$
    SWAP (q_{i+1}, q_j)
    SWAP(q_{k+1}, q_l)
    L = L $\setminus\{q_i, q_{i+1}, q_k, q_{k+1}\}$
  else
    // $\exists$ CNOT$(q_{\frac{n}{2}}, q_l)$ with $l > \frac{n}{2}$
    // and $\exists$ CNOT(q_i, q_{l-1}) with $i < \frac{n}{2}$
    SWAP $(q_{\frac{n}{2}}, q_{l-1})$
    SWAP $(q_i, q_{\frac{n}{2}-1})$
    L = L $\setminus\{q_{\frac{n}{2}-1}, q_{\frac{n}{2}}, q_{l-1}, q_l\}$
  end if
  if $L \neq \emptyset$ **then**
    Sort(L)
  end if
end function

2.3.7 Quantum Processor

A quantum processor, which corresponds to individual processor in a classical distributed system, has several qubits and links between two qubits in a limited topology.

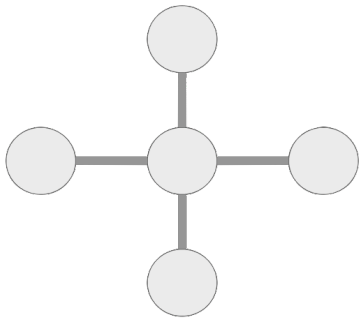


Figure 2.10: An example of the layout of a quantum processor

Usually, qubits in a current quantum processor are connected with a few neighboring qubits due to its physical restriction on a hardware. Also, the error rate of a CNOT gate is much higher than that of a single quantum gate.

2.3.8 Communication Link for Distributed Quantum Computing

Networking between two quantum processors need two types of communication links, which are classical links that transmit measurement results in the process of either telegate or teledata non-local CNOT gate, and quantum links that prepare create entanglement between the two qubits between the two neighboring quantum processors, or communication qubits of both processors.

Chapter 3

Related Work

3.1 Performance of An Interprocessor CNOT Gate

Both the execution time and required amount of resource for inter-node communication are important because it significantly affects the total execution time and the whole architecture of a distributed quantum computing system.

The work [22] proved that one bit of classical communication in each direction and one bell pair are sufficient for the non-local CNOT gate. It also proposes the optimal implementation of a non-local CNOT in terms of communication overhead and the number of required quantum gates.

The work [23] compared the performance between "telegate" and "teledata" approach in terms of various number of data qubits, communication qubits, and its network topology. It presents the fact that the teledata approach is faster than the telegate approach, and that decomposition of a quantum gate will improve the performance. It also shows that each node should have a few logical qubits and two communication qubits.

3.2 Minimization of the Number of Interprocessor Communication

Dividing the given quantum circuit into several fragments one of the main jobs for a distributed quantum compiler. In the process of partitioning the given quantum circuit, the distributed quantum compiler has to minimize the number of inter-processor communication in order to reduce the delay in the entire circuit execution.

Severals algorithm for finding the quantum circuit partition with minimum number of inter-processor communications have been proposed and these algorithms are based on exhaustive search [26], graph partitioning [27], hypergraph partitioning [28], genetic algorithm [29], dynamic programming [30], window-based quantum circuit partitioning [31] , and connectivity matrix of the quantum circuit [32].

3.3 Quantum Compiler For Distributed Quantum Computing

Ferrari *et al.* [25] discussed the design of a general-purpose, efficient, and effective distributed quantum computer. General purpose means no assumption about the given quantum circuit. Efficient means polynomial time complexity that grows polynomially with the number of qubits and linearly with the circuit depth. Effective assures a polynomial worst-case overhead in terms both depth of the compiled circuit, the number of entanglement generation.

This study also derived the analytical upper bound of the circuit depth both for the entanglement-based non-local operation and the data-qubit-swapping-based non-local operation.

The depth overhead of the entanglement-swapping-based strategy would be at most

$$\frac{n}{2}\,\mathrm{d_{es}} \tag{3.1}$$

$$\mathrm{d_{es}} = \mathrm{c_{le}} + \mathrm{c_{bsm}} + \mathrm{c_{cx}} \tag{3.2}$$

.

On the other hand, the depth overhead of the data-qubit-swapping strategy is

$$\frac{n}{4}\,\mathrm{d_{qs}} + \mathrm{d'_{qs}} \tag{3.3}$$

$$\mathrm{d_{qs}} = 3(\mathrm{c_{le}} + \mathrm{c_{bsm}} + \mathrm{c_{cx}}) \tag{3.4}$$

$$\mathrm{d'_{qs}} = \mathrm{c_{le}} + \mathrm{c_{cx}} \tag{3.5}$$

.

n is the number of qubits, c_{le} is the number of layers required to perform the link entanglement, c_{bms} is that to perform entanglement swapping, c_{cx} is that to perform remote CNOTs. This study mentions that parameters c_{le}, c_{bms}, c_{cx} heavily depends on the underlying hardware architecture.

It also compare the performance of both strategies with the previous work [28]. It experimentally demonstrated that the entanglement-swapping-based strategy requires less number of layers for link generation, and the data-qubit-swapping-based strategy requires less circuit depth, on the worst network topology (the linear topology with one qubit on an each processor).

3.4 Toward Experimental Realization of Distributed Quantum Computing

Jiang *et al.*[33] proposed a method to perform distributed quantum computing even by using 5-qubit noisy processors and noisy quantum links.

Nickerson *et al.*[34] proposed a protocol to perform distributed quantum computing which utilizes topological codes to perform purification in the processor and also states that this protocol works if the error rate of quantum links is less than 10% and that of quantum processor for initialization, state manipulation, and measurement is less than 0.82%

Oi *et al.*[35] proposed an architecture for distributed, fault-tolerant quantum computing that enables scalable that achieves scalable quantum error correction.

Daiss *et al.*[36] experimentally achieved a remote quantum gate and successfully implemented four Bell states.

Kim *et al.*[37] proposed an architecture for a large-scale quantum computing that combines ion-trap and optical technologies.

LaRacuente *et al.*[38] proposed an architecture of a superconducting chiplet that connects the physical processor and microwave links.

Van Meter *et al.*[39] discusses required technology stack for distributed quantum computing in terms of various hardware technologies and software that manipulates these technologies.

Chapter 4

Problem Definition and Proposal

This chapter explains a new problem about the qubit allocation in the distributed quantum computing system and propose the solution for that problem.

4.1 Problem Definition

In order to execute distributed computing, a collection of tasks should be allocated onto multiple processors limited by a certain network topology. These processors might have different properties such as their execution time and their memory size. Algorithms regarding distributed computing aim to achieve efficiency in terms of either reduction in the total execution time or increase of total reliability compared to the case of execution of the same tasks on the single processor.

Previous works about qubit allocation on a single quantum processor tries to improve the fidelity of the output quantum state even in the presence of physical noise on a hardware, which is "reliability" for quantum computing, and these works can be directly applied to improve the total fidelity even in the case of distributed quantum computing if the whole quantum computing cluster is considered as a single larger-scale quantum processor .

However, qubit allocation for distributed quantum computing to minimize the total execution time has not been investigated, even though this is one of the two major optimization criteria for task allocation problem in the classical setting.

This work formulates the problem of qubit allocation on multiple quantum processors with varying execution time as an optimization problem and define its objective function. It also demonstrates the validity of the proposed method by numerical simulation, which adopted simulated annealing as a heuristic optimization algorithm.

4.2 Formulation as An Optimization Problem

Suppose a distributed quantum computing system consists of N quantum processors connected via communication links. Each quantum processor has limited number of qubits and execution time.

A quantum circuit in the program consists of several qubits and M gates, including CNOTs which corresponds to an interaction graph $G(V, E)$. V represents a set of qubits and E represents set of two qubits involved in each CNOT gate. $q_i \in V$ is labeled by the qubit index, and $(\text{control}, \text{target}) \in E$ is labeled by control-target relationship of all the CNOT gates.

The problem is how to allocate each qubits in the given quantum circuit to which processor with varying execution time in order to minimize the total execution time. This problem can be formulated as an optimization problem, which requires a cost function, which is the value to either maximize or minimize to acquire the optimal solution.

4.3 Objective Function

Suppose A be the optimal assignment such that $A(q_i) = p_j$ if a qubit q_i in the given quantum circuit to a quantum processor p_j. Qubits allocated to a quantum processor p_j is denoted as qubits_j, single qubit gates allocated to a qubit q_i is $gates_i$ and the execution time on a quantum processor q_j is time_j.

The cost of executing all single-qubit gates on a qubit q_i on a quantum processor p_j is

$$\sum_{\text{gate} \in \text{gates}_\text{i}} \text{time}_\text{j} \tag{4.1}$$

.

Therefore, the cost of executing all the single-qubit gates on all the qubits allocated on quantum processor p_j is

$$\text{GATECOST}_\text{j} = \sum_{\text{qubit} \in \text{qubits}_\text{j}} \sum_{\text{gate} \in \text{gates}_\text{qubit}} \text{time}_\text{j} \tag{4.2}$$

.

Suppose a CNOT gate CNOT(control, target) involves two qubits, which are control $\in qubits_s$ and target $\in \text{qubits}_\text{t}$ and all the CNOT gates in a quantum processor p_j are denoted as CNOTs_j.

The communication cost in a quantum processor p_j is

$$\text{COMMCOST}_\text{j} = \sum_{\substack{\text{CNOT(control,target)} \in \text{CNOTs}_\text{j} \\ \text{control} \in \text{q}_\text{s} \\ \text{target} \in \text{q}_\text{t}}} \text{path\_length}(s, k) \tag{4.3}$$

.

path _length is the length of the path between the processor s and the processor t on the given network topology, and the processor j is same as at least either the processor s or the processor t.

Thus, the total cost on a quantum processor p_j is

$$\text{COST}_\text{j} = \text{GATECOST}_\text{j} + \text{COMMCOST}_\text{j} \tag{4.4}$$

.

Both execution of single qubit gates and communication with other processors affect the total execution time on each processor, and because the processor with the greatest

cost will decide the total execution time on the whole distributed quantum system, the following value should be calculated.

$$\text{MAXCOST} = \max\{\text{COST}_\text{j} \,|1 \leq j \leq N\} \tag{4.5}$$

.

Also, minimizing this value will reduce both the execution time for quantum gates execution and interprocessor communication, and the objective function of this problem is

$$\min \text{MAXCOST} \tag{4.6}$$

.

In the chapter 5, the optimization of this proposed objective function is performed by one of the most popular heuristic optimization algorithm called simulated annealing, which will be explained in the following section.

4.4 Simulated Annealing

Simulated annealing is a heuristic algorithm which reaches to the global optimal solution in some cases [40, 41]. Its idea comes from cooling the molten metal until it gains crystal structure, so it calculates the values of "temperature", which how long the optimization has been executed and "energy" which evaluates how close the current answer is to the optimal solution. The algorithm starts with high temperature and energy, and as the temperature becomes lower, the solution changes randomly and the combination and the answer after randomization process is accepted even its energy becomes higher than its previous answer in order to avoid being stuck in the local minimum.

Here is the pseudocode for the simulated annealing algorithm.

Algorithm 2 Simulated Annealing

Input: A random allocation A, temperature T, iteration number IterNum
Output: The optimal allocation A'

```
1: function SimulatedAnnealing(A, T, IterNum)
2:     A' = A
3:     for iter := 1 to IterNum do
4:         temp := T ∗(1 − iter/IterNum)
5:         copyA ← copy(A')
6:         newA ← move(copyA)
7:         eng ← calc_energy(copyA)
8:         neweng ← calc_energy(newA)
9:         if accept_prob(eng, neweng, temp) > randomvalue(0, 1) then
10:            A' = NewA
11:        end if
12:    end for
13:    return A'
14: end function
```

Algorithm 3 Finding a neighbor state

Input: Processor list P $\{P_0, P_1, \ldots P_N\}$, initial allocation A $\{P_0 : qubits_0, P_1 : qubits_1, \ldots P_n : qubits_n\}$
Output: New allocation A

```
1: function MOVE
2:     P_i ← a randomly selected processor
3:     P_j ← another randomly selected processor
4:     qindex_i ← a randomly selected qubit index from 0 to len(qubits_i)
5:     qindex_j ← a randomly selected qubit index from 0 to len(qubits_j)
6:     A[P_i][qindex_i], A[P_j][qindex_j] = A[P_j][qindex_j], A[P_i][qindex_i]
7: end function
```

Algorithm 4 Calculating the energy value

Input: initial allocation A $\{P_0 : qubits_0, P_1 : qubits_1, \ldots P_n : qubits_n\}$, a quantum gate list gate_list $\{gate_0, \ldots, gate_N\}$, an execution time list time_list [time_0, ... time$_N$], network topology N

Output: An energy value E

```
1: function CALC_ENERGY(A, gate_list)
2:     processor_list ← [keys in A]
3:     gate_cost_list ← [0 for processor in processor_list]
4:     comm_cost_list ← [0 for processor in processor_list]
5:     for processor_id := 0 to length of processor_list −1 do
6:         for gate := gate_0 to gate_N do
7:             if gate_name ≠ CNOT and gate_index ∈ A[processor_id] then
8:                 gate_cost_list[processor_id] += time_list[processor_id]
9:             end if
10:        end for
11:    end for
12:    distance_matrix ← N_distance_matrix
13:    for processor_id := 0 to len(processor_list) −1 do
14:        for gate := gate_0 to gate_N do
15:            if gate_name = CNOT then
16:                if gate_index, gate_target_index ∈ A[processor_id] then
17:                    comm_cost_list[processor_id] += 0
18:                else if gate_index ∈ A[processor_id] then
19:                    for processor'_id := 0 to length of processor_list −1 do
20:                        if gate_target_index ∈ A[processor'_id] then
21:                            distance ← distance_matrix[processor_id][processor'_id]
22:                            comm_cost_list[processor_id] += distance
23:                        end if
24:                    end for
25:                else if gate_gate_index ∈ A[processor_id] then
26:                    for processor'_id := 0 to length of processor_list −1 do
27:                        if gate_index ∈ A[processor'_id] then
28:                            distance ← distance_matrix[processor'_id][processor_id]
29:                            comm_cost_list[processor_id] += distance
30:                        end if
31:                    end for
32:                end if
33:            end if
34:        end for
35:    end for
36:    for processor_id := 0 to length of processor_list −1 do
37:        gate_cost ← gate_cost_list[processor_id]
38:        comm_cost ← comm_cost_list[processor_id]
39:        cost_list[processor_id] ← gate_cost + comm_cost
40:    end for
41:    return max cost_list
42: end function
```

Algorithm 5 Calculating the acceptance probability

Input: current energy value cur_eng, new energy value new_eng, current temperature temp

Output: an acceptance probability *prob*

```
1: function ACCEPT_PROB(cur_eng, new_eng, temp)
2:     if cur_eng < new_eng then
3:         return 1
4:     else
5:         return exp (-(new_eng-cur_eng)/temp)
6:     end if
7: end function
```

Chapter 5

Heqsim: Heterogeneous Quantum Computing Simulator

5.1 About

Heqsim is a distributed quantum computing simulator that the author is developing, and will be an open-source Python library after some minor bugs will be fixed (The GitHub repository is private right now). Currently, this simulator has the following features such as

- Users can specify different execution time per gate to each quantum processor
- Users can also specify the network topology among quantum processors

. Heqsim is the first distributed quantum computing simulator that have these features. The existing distributed quantum computing simulator NetQuil [42] and Inter-linq [43] , both of which are only restricted to fully-connected network topology.

5.2 How Heqsim works

Here is the procedure of the simulation using heqsim.

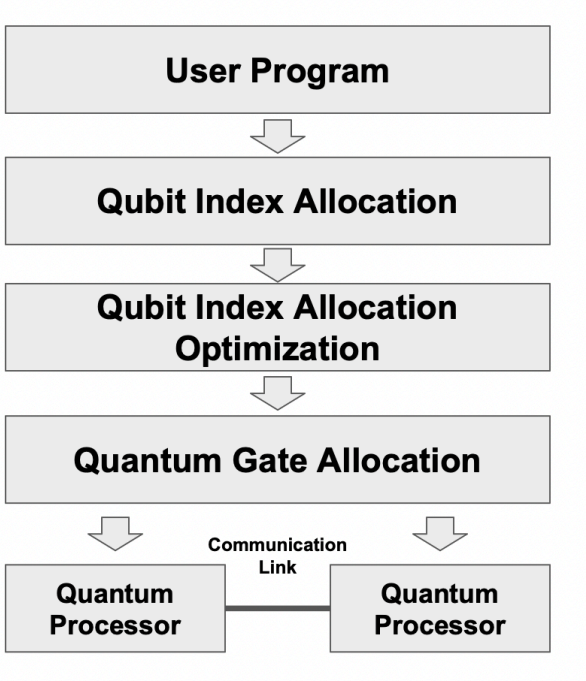


Figure 5.1: Procedure of heqsim simulation

.

5.2.1 User Program

Users can specify the details (number of qubits, execution time) of each quantum processor and the network topology of the entire quantum computing cluster. Also, they can specify the quantum circuit that they want to implement.

5.2.2 Qubit Index Allocation

Heqsim receives the information of the total qubits in the given quantum circuit and distributes each qubit to random quantum processors.

5.2.3 Qubit Index Allocation Optimization

The Qubit Allocation Optimizer optimized the qubit index allocation by using the method described in the previous chapter.

5.2.4 Quantum Gate Allocation Optimization

The module called Gate Allocator distributed quantum gates according to the result of qubit index allocator and also decomposes some CNOT gates according to the network topology.

5.2.5 Quantum Processor

Quantum Processor applies actual quantum gates given by the gate allocator. The single gates are implemented by parallel execution, but communication is emulated by communication links between two quantum processors.

Chapter 6

Evaluation

This chapter investigates the efficiency of the allocation method proposed in the previous chapter, under the distributed quantum computing system with several processors with different execution time in a limited network topology.

6.1 Settings

In this experiment, three 2-qubit quantum processors with different execution time are connected in a linear topology.

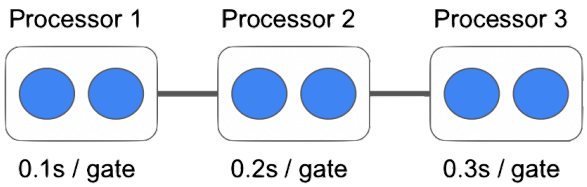
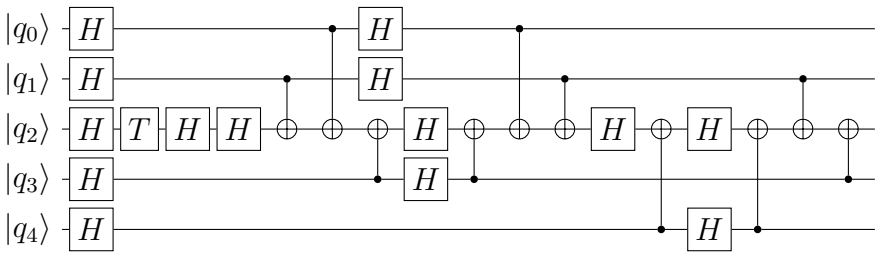


Figure 6.1: The details of each quantum processor

.

The author executed the following circuit on these three quantum processors. This circuit is a quantum circuit for encoding 5-qubit quantum repetition code, which is one of the circuits used for benchmarking purposes [44].



.

6.2 Result

This experiment was executed using the distributed quantum computing simulator heqsim (Heterogeneous quantum computing simulator) [45]. The author compared the total execution time of four different allocation cases, which are

- when qubits are randomly allocated
- when only the gate execution cost is optimized
- when only the communication cost is optimized
- when both costs are optimized

.

I measured the total execution time in the four cases, ten trials for each, and compared the average execution time. Here is the experiment result.

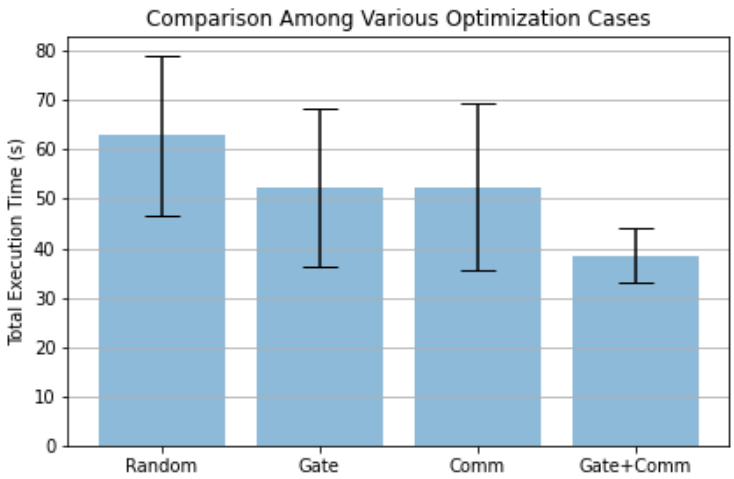


Figure 6.2: Result

.

The figure shows that the total execution time becomes the shortest when the both gate execution cost and communication cost are optimized.

Chapter 7

Conclusion

7.1 Discussion

This thesis aims to propose an effective scheme for qubit allocation for distributed quantum computing to reduce the total execution time, and the chart in the previous chapter clearly demonstrates that the case when the both gate execution cost and communication cost are optimized performs the best. This fact also states that, similar to the task allocation algorithm for classical distributed computing, people have to take both the task (quantum gate) execution cost and communication cost into account in order to come up with an (nearly-) optimal qubit allocation in terms of reducing the total execution time.

In this experiment, the total execution time of the gate-cost-based case and that of the communication-cost-based-case are almost the same. However, there are some cases where either the gate-cost-based optimization or the communication-cost-based optimization works better than the other. For example, gate-cost-based optimization would work better if the given circuit have more single qubit gates than CNOT gates and these gates are fairly allocated to each qubit. On the other hand, the communication-cost-based optimization yields a better performance if the given quantum circuit has more CNOT gates than one-qubit gates, and each processor has less neighboring processors, such as linear topology.

7.2 Significance of This Work

This work has two main significance for the field of distributed quantum computing. One is that the scheme proposed in this thesis is the first execution-time-oriented qubit allocation algorithm for distributed quantum computing. If the mankind finally overcomes the problem of severe physical noise on each quantum hardware and achieves the large-scale, fault-tolerant, and even distributed quantum computing, they may want to accelerate the whole computing process to achieve more efficient quantum computing. In order to speed up the execution of a large-scale quantum circuit with enormous amount of quantum gates on heterogeneous quantum processors, they have to care about both the execution time on each quantum processor and communication cost between each pair of

those processors.

The other significance is that the qubit allocation with the minimum amount of communication is, at least, not always optimal in terms of its total execution time, as I stated in the previous section.

7.3 Future Work

This work only focuses on deciding which qubit should be allocated to which quantum processor, but a few other procedures have to be executed in order to achieve more efficient quantum computing. For example, after qubits are allocated to each processor, the quantum circuit should be modified to reduce the number of communication between quantum processors. Also, similar to the qubit mapping schemes for local quantum processors, the algorithm for choosing which allocated qubit should be mapped to which physical qubit should also be investigated.

Bibliography

[1] RichardP Feynman. Simulating physics with computers. *International Journal of Theoretical Physics*, 21(6-7):467–488, June 1982.

[2] Peter W Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM review*, 41(2):303–332, 1999.

[3] Seth Lloyd. Universal quantum simulators. *Science*, 273(5278):1073–1078, 1996.

[4] A. A. Houck, Jens Koch, M. H. Devoret, S. M. Girvin, and R. J. Schoelkopf. Life after charge noise: recent results with transmon qubits. *Quantum Information Processing*, 8(2-3):105–115, Feb 2009.

[5] Boris B Blinov, Dietrich Leibfried, C Monroe, and David J Wineland. Quantum computing with trapped ion hyperfine qubits. *Quantum Information Processing*, 3(1):45–59, 2004.

[6] F. Jelezko and J. Wrachtrup. Single defect centres in diamond: A review. *phys. stat. sol. (a)*, 203(13):3207–3225, 2006.

[7] Craig Gidney and Martin Ekerå. How to factor 2048 bit rsa integers in 8 hours using 20 million noisy qubits. *Quantum*, 5:433, Apr 2021.

[8] Lov K Grover. Quantum telecomputation. *arXiv preprint quant-ph/9704012*, 1997.

[9] Marcos Yukio Siraichi, Vinícius Fernandes dos Santos, Caroline Collange, and Fernando Magno Quintao Pereira. Qubit allocation. In *Proceedings of the 2018 International Symposium on Code Generation and Optimization*, CGO 2018, page 113–125, New York, NY, USA, 2018. Association for Computing Machinery.

[10] Prakash Murali, Jonathan M. Baker, Ali Javadi-Abhari, Frederic T. Chong, and Margaret Martonosi. Noise-adaptive compiler mappings for noisy intermediate-scale quantum computers. In *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS '19, page 1015–1029, New York, NY, USA, 2019. Association for Computing Machinery.

[11] Thomas Häner, Damian S. Steiger, Torsten Hoefler, and Matthias Troyer. Distributed quantum computing with QMPI. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '21, New York, NY, USA, 2021. Association for Computing Machinery.

[12] Alwin Zulehner, Alexandru Paler, and Robert Wille. An efficient methodology for mapping quantum circuits to the ibm qx architectures, 2018.

[13] Ajay D Kshemkalyani and Mukesh Singhal. *Distributed computing: principles, algorithms, and systems*. Cambridge University Press, 2011.

[14] Yichuan Jiang, Yifeng Zhou, and Wanyuan Wang. Task allocation for undependable multiagent systems in social networks. *IEEE Transactions on Parallel and Distributed Systems*, 24(8):1671–1681, 2013.

[15] Perng-Yi Richard Ma, Lee, and Tsuchiya. A task allocation model for distributed computing systems. *IEEE Transactions on Computers*, C-31(1):41–47, 1982.

[16] Gamal Attiya and Yskandar Hamam. Task allocation for maximizing reliability of distributed systems: A simulated annealing approach. *Journal of Parallel and Distributed Computing*, 66(10):1259–1266, 2006.

[17] V.M. Lo. Heuristic algorithms for task assignment in distributed systems. *IEEE Transactions on Computers*, 37(11):1384–1397, 1988.

[18] Gamal Attiya and Yskandar Hamam. Task allocation for minimizing programs completion time in multicomputer systems. volume 3044, pages 97–106, 05 2004.

[19] Vasil S. Denchev and Gopal Pandurangan. Distributed quantum computing: A new frontier in distributed systems or science fiction? *SIGACT News*, 39(3):77–95, sep 2008.

[20] Anocha Yimsiriwattana and Samuel J Lomonaco Jr. Distributed quantum computing: A distributed shor algorithm. In *Quantum Information and Computation II*, volume 5436, pages 360–372. International Society for Optics and Photonics, 2004.

[21] Iaakov Exman and Efrat Levy. Quantum probes reduce measurements: Application to distributed grover algorithm. arXiv:1208.3905, 08 2012.

[22] Jens Eisert, Kurt Jacobs, Polykarpos Papadopoulos, and Martin B Plenio. Optimal local implementation of nonlocal quantum gates. *Physical Review A*, 62(5):052317, 2000.

[23] Rodney Van Meter, WJ Munro, Kae Nemoto, and Kohei M Itoh. Arithmetic on a distributed-memory quantum multicomputer. *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, 3(4):1–23, 2008.

[24] Charles H. Bennett, Gilles Brassard, Claude Crépeau, Richard Jozsa, Asher Peres, and William K. Wootters. Teleporting an unknown quantum state via dual classical and einstein-podolsky-rosen channels. *Phys. Rev. Lett.*, 70:1895–1899, Mar 1993.

[25] Davide Ferrari, Angela Sara Cacciapuoti, Michele Amoretti, and Marcello Caleffi. Compiler design for distributed quantum computing. *arXiv preprint arXiv:2012.09680*, 2020.

[26] Mariam Zomorodi-Moghadam, Mahboobeh Houshmand, and Monireh Houshmand. Optimizing teleportation cost in distributed quantum circuits. *International Journal of Theoretical Physics*, 57(3):848–861, 2018.

[27] Omid Daei, Keivan Navi, and Mariam Zomorodi-Moghadam. Optimized quantum circuit partitioning. *International Journal of Theoretical Physics*, 59(12):3804–3820, 2020.

[28] Pablo Andres-Martinez and Chris Heunen. Automated distribution of quantum circuits via hypergraph partitioning. *Physical Review A*, 100(3):032308, 2019.

[29] Mahboobeh Houshmand, Zahra Mohammadi, Mariam Zomorodi, and Monireh Houshmand. An evolutionary approach to optimizing teleportation cost in distributed quantum computation. *International Journal of Theoretical Physics*, 59, 04 2020.

[30] Zohreh Davarzani, Mariam Zomorodi-Moghadam, Mahboobeh Houshmand, and Mostafa Nouri-baygi. A dynamic programming approach for distributing quantum circuits by bipartite graphs. *Quantum Information Processing*, 19(10):1–18, 2020.

[31] Eesa Nikahd, Naser Mohammadzadeh, Mehdi Sedighi, and Morteza Saheb Zamani. Automated window-based partitioning of quantum circuits. *Physica Scripta*, 96, 12 2020.

[32] Ismail Ghodsollahee, Zohreh Davarzani, Mariam Zomorodi, PawfB QBawiak, Monireh Houshmand, and Mahboobeh Houshmand. Connectivity matrix model of quantum circuits and its application to distributed quantum circuit optimization. *Quantum Inf. Process.*, 20:1–21, 2021.

[33] Liang Jiang, Jacob M Taylor, Anders S Sørensen, and Mikhail D Lukin. Scalable quantum networks based on few-qubit registers. *International Journal of Quantum Information*, 8(01n02):93–104, 2010.

[34] Naomi H Nickerson, Ying Li, and Simon C Benjamin. Topological quantum computing with a very noisy network and local error rates approaching one percent. *Nature communications*, 4(1):1–5, 2013.

[35] Daniel KL Oi, Simon J Devitt, and Lloyd CL Hollenberg. Scalable error correction in distributed ion trap computers. *Physical Review A*, 74(5):052313, 2006.

[36] Severin Daiss, Stefan Langenfeld, Stephan Welte, Emanuele Distante, Philip Thomas, Lukas Hartung, Olivier Morin, and Gerhard Rempe. A quantum-logic gate between distant quantum-network modules. *Science*, 371(6529):614–617, 2021.

[37] Jungsang Kim and Changsoon Kim. Integrated optical approach to trapped ion quantum computation. *arXiv preprint arXiv:0711.3866*, 2007.

[38] Nicholas LaRacuente, Kaitlin N. Smith, Poolad Imany, Kevin L. Silverman, and Frederic T. Chong. Short-range microwave networks to scale superconducting quantum computation, 2022.

[39] Rodney Van Meter and Simon J Devitt. Local and distributed quantum computation. *arXiv preprint arXiv:1605.06951*, 2016.

[40] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, 1983.

[41] Emile H. L. Aarts and Jan H. M. Korst. Simulated annealing and boltzmann machines - a stochastic approach to combinatorial optimization and neural computing. In *Wiley-Interscience series in discrete mathematics and optimization*, 1990.

[42] Matthew Radzihovsky and Zachary Espinosa. netquil: A quantum playground for distributed quantum computing simulations. *Bulletin of the American Physical Society*, 65, 2020.

[43] Rhea Parekh, Andrea Ricciardi, Ahmed Darwish, and Stephen DiAdamo. Quantum algorithms and simulation for parallel and distributed quantum computing. *arXiv preprint arXiv:2106.06841*, 2021.

[44] Ang Li, Samuel Stein, Sriram Krishnamoorthy, and James Ang. Qasmbench: A low-level QASM benchmark suite for NISQ evaluation and simulation, 2021.

[45] Makoto Nakai. heqsim: Heterogeneous Quantum Computing Simulator, 4 2022.

[46] M. D. Weerdt, Y. Zhang, and Tomas Klos. Multiagent task allocation in social networks. *Autonomous Agents and Multi-Agent Systems*, 25:46–86, 2011.

Acknowledgment

First and foremost, I would like to show my gratitude to Professor Van Meter for offering me with opportunities to work on various research fields including quantum machine learning, and digital quantum simulation and distributed quantum computing, and make so many mistakes in the first three years. I could not have felt confident about progress of my knowledge and overall capabilities in the last four years if I join in any other research groups.

I also feel grateful to Professor Takahiko Satoh, who have provided me with severe and honest feedbacks about my slides and the way I make a presentation. Without his patient support, I would not have been able to express my research topic using easy words to those who are not familiar with quantum computing.

I would also like to appreciate Professor Michal Hajdušek, who allowed me to work on a digital quantum simulation about quantum synchronization between two spin-1 particles. Even though understanding the setting itself is a really hard work, it is my honor to work on quantum physics, which is my long-time favorite since when I was in high school. I will resume this joint research after I start my Master's.

I also appreciate Professor Masashi Aono, who taught me the basics of optimization theory and complex system in his lectures. I would not have thought about working on combinatorial optimization for my graduation thesis, and quantum synchronization for the other research, if I didn't take his intriguing lectures.

I also thank other members in AQUA, especially Ryosuke Satoh for mentoring my research work in this entire year and taking care of me in various aspects, Naphan Benchasattabuse for always giving me critical feedbacks after my progress report, Yasuhiro Okura for suggesting me the details of this research topic, Taaki Matsuo for treating me so many times, Nozomi Tanetani for keeping encouraging me when I felt devastated in the Mitou Target, and Shigetora Miyashita for having a thoughtful conversation about modern physics.

Last but not least, I would like to express my deepest gratitude to my family members who have raised me and become supportive in any situations.